

Scalability and Resource Usage of an OLAP Benchmark on Clusters of PCs

Michela Taufer, Thomas Stricker, Roger Weber

Department of Computer Science
ETH Zentrum
CH-8092 Zurich, Switzerland
taufer,tomstr@acm.org
weber@inf.ethz.ch

ABSTRACT

Designing clusters of PCs for distributed databases processing OLAP (On Line Analytical Processing) workloads in parallel with good scalability remains a particular challenge as we are lacking a deep understanding of the architectural issues around resource usage by standard DBMSs on distributed platforms.

To address this problem, we present a novel performance monitoring framework for filtering and abstracting samples of performance data from low level counters into a high level performance picture. Our framework is used side by side with the DBMS and delivers many interesting insights about the most critical resource in the different queries and systems configuration. As required for a larger distributed hardware/software system, our solution comprises software instrumentation at the OS level, tools for gathering performance relevant data and an analytical model for performance evaluation and performance prediction to future platforms.

We demonstrate the viability of our approach with the in-depth analysis of distributed TPC-D, a standard OLAP benchmark running on clusters of commodity PCs. Based on the data provided by our framework, we isolate and resolve a few crucial performance issues of OLAP workloads on clusters. For different queries, we give a workload characterization in terms of resource usage, quantify the optimal scalability and investigate the impact of the networking speed on the overall application performance. We show that the disk performance and CPU speed remains the most critical resource bottleneck for most queries. Queries with a lot of inter-node communication are limited by the communication software inefficiency within the DBMS and not by the raw networking speeds. A systematic performance evaluation constitutes a solid basis for architectural decisions and system optimization in clusters of PCs that are dedicated to large parallel database systems.

Categories and Subject Descriptors

C.4 [PERFORMANCE OF SYSTEMS]: Measurement techniques, Modeling techniques; H.2.4 [DATABASE MANAGEMENT]: Systems—*Distributed databases*.

General Terms

Measurement, Performance.

Keywords

Parallel databases, distributed OLAP processing, cluster of PCs, performance analysis, workload characterization.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SPAA'02, August 10-13, 2002, Winnipeg, Manitoba, Canada.
Copyright 2002 ACM 1-58113-529-7/02/0008 ...\$5.00.

1. INTRODUCTION

Clusters of commodity PCs have become a highly popular platform for distributed computing since simple PC workstation have one of the best price performance ratios in the market for computing equipment. Moving large OLAP (On Line Analytical Processing) database workloads to clusters of commodity PCs looks very tempting because of the lower cost of clusters and the significant potential for speed-up. As any other effort in parallel and distributed computing, the success depends on proper performance analysis of the entire system including hardware and software. Engineering such systems for scalability and good performance remains a highly difficult task, since we are lacking tools and instrumentation for performance analysis that work in distributed systems and with standard DBMS.

The scalability of the different TPC-D queries, chosen as a typical example of OLAP applications on PC clusters presents a picture of unpredictable performance and speed-up numbers that seem to be highly sensitive to the nature of the workload. Figure 1 shows a typical study case for the TPC-D Benchmark distributed with TP-Lite (a software tool for distributing queries) on three nodes of a cluster of PCs. We expect a speed-up of almost three (shown with the

upper line in the picture) however, not all the queries are sped up (i.e 2,9 and 10) and some of them (i.e. 5 and 7) do not even end in a reasonable time due to some inefficiency in parallel processing.

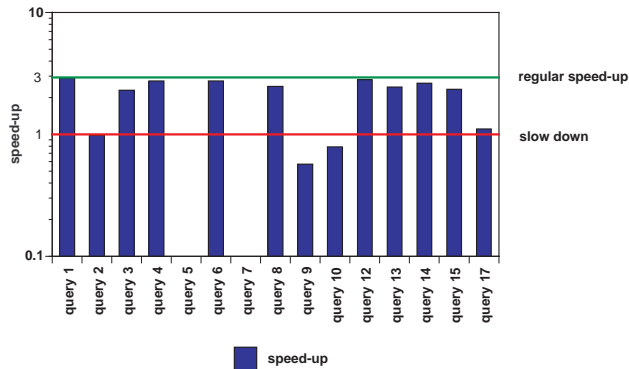


Figure 1: Scalability for the TPC-D Benchmark distributed across three nodes of a cluster of commodity PCs with TP-Lite

For studying the reason of suboptimal speed-up in Figure 1, most DBMS like ORACLE incorporate elaborate instrumentation for performance monitoring and tuning [9, 6]. However, such instrumentation normally works on the basis of data collection inside the DBMS and not on the basis of mapping the operating system state or the basic performance monitoring data back to an abstraction level that is more appropriate for a user. In particular, the performance monitoring instrumentation of ORACLE only accounts for the total count of operations and does neither allow to efficiently sample performance counts at certain intervals nor allow for this information to be collected efficiently from a large number processing nodes in a cluster. While some of the logical or table access counts would certainly be interesting, most performance information of the database management system does not directly relate to the usage of physical resources in a distributed system and is therefore hard to use in a framework that aims at predicting the execution time based on application and platform parameters.

In applications that use multiple processors and distribute a large workload for the sake of higher execution speed, precise understanding of resource utilization becomes a crucial concern for finding the causes of performance and scalability problems. Such an investigation requires a proper engineering of the complex system. A precise understanding of resource utilization is not just required for fine tuning a running system, but also for the prediction of scalability or the study of viability of the application on new, alternative platforms such as clusters of low cost commodity PCs. The problem of understanding which hardware component leads to which part of the total execution time in simple queries has been previously studied in [1] for a memory resident database and different kind of commercial DBMSs running on an Intel Xeon and Windows NT 4.0. The study focuses on processor and memory interactions excluding effects of I/O subsystems and data partitioning.

Our main contribution in this paper is to explain precise resource usage and the scalability of the TPC-D Benchmark running on clusters of PCs including disks and networks. We demonstrate that there is indeed a systematic way to

filter the raw performance data provided by the operating system and turn the result into a more abstract performance picture that reflects the resource usage of the distributed application code. As cluster architects we can improve the configuration of future PC clusters based on this data about resource usage. Depending on the most performance critical resources we can improve the cost/performance ratio by using cheaper or more expensive CPUs, disks, memory systems (motherboard) or interconnects between the nodes. If we can use cheaper components without penalty in performance we can afford a larger number on nodes in a cluster. Since cluster computing is always about the best use of commodity components, we look at parallel- and distributed systems built entirely with commodity hardware and commercial software components. In particular, we combine the open source operating system LINUX with the proprietary single node DBMS of ORACLE and a lean experimental software layer for the distributing processing of queries. The resulting high performance database system for decision support workloads can be classified as a shared-nothing architecture. Since we are interested in using cluster for very large data sets exceeding the disk capacities of a single node our data distribution scheme relies on partitioning rather than on replication.

The rest of the paper is organized as follows: Section 2 describes our approach to distribute the OLAP workload to the nodes of a PC cluster based on the TP-Lite system and the basic characteristics TPC-D benchmark. Section 3 presents our performance monitoring framework for distributed systems using raw performance data supplied by the operating system. Section 4 applies our methodology to three performance issues in distributed databases. We use the framework to characterize the workload of TPC-D and classify different queries according to their resource usage, to investigate the scalability problems of certain queries and to determine the performance impact of different interconnect speeds found in modern PC clusters. Section 6 concludes the paper with a statement about the viability of our approach to instrument a complex system of standard software properly for performance analysis and performance optimization.

2. DISTRIBUTED TPC-D BENCHMARK ON PC CLUSTERS

In this Section, we give a brief overview on TP-Lite [3], the software for distributing a single TPC-D [21] query across our cluster. Furthermore, we discuss the most important characteristics of the hardware- and software platforms used. The Section concludes with the definition of the factor- and the parameter space of the benchmark for workload characterization.

2.1 Distributing OLAP Workloads on PCs Clusters

Our performance analysis method was motivated strongly by OLAP (On-Line Analytic Processing) applications running in parallel on a cluster of PCs. Although the PowerDB project at ETH Zurich deals mainly with parallel OLTP (On-Line Transaction Processing, e.g. TPC-C [20]), our interest is more in OLAP workloads (e.g. TPC-D [21]). OLAP applications deal with large quantities of data in a single query that could potentially benefit from high speed interconnects in advanced PC clusters. Furthermore, high perfor-

mance workloads in OLTP do require a very large number of simultaneous queries to scale, and - in general - do not result in large data transfers. Therefore, large scale OLTP setups are much harder to generate than a large scale OLAP job, and less interesting for computer architects. The accurate behavior of database workload for OLTP jobs is addressed for shared-memory multiprocessors in [14].

We still use the TPC-D [21] benchmark as a representative of OLAP applications for historical reasons but could easily migrate our approach to TPC-H or TPC-R and the qualitative aspects of this article would remain the same if these more recent benchmarks were used. Some of the newer work in the PowerDB project also deals with TPC-H and TPC-R also including concurrent updates.

2.2 TP-Lite Approach

In principle, a parallel implementation of a database on a cluster of PCs works as follows: clients send their requests (i.e. SQL-transactions) to a so-called coordinator. The coordinator analyzes these requests, partitions them into several independent sub-transactions, and routes the sub-transactions to the nodes within the cluster. The goal of the partitioning and the routing is to minimize response times of queries and to maximize the throughput of SQL transactions. While the general approach is rather complex due to update operations and concurrency, we focus our attention to a query-only environment as this is sufficient for most work in OLAP. In this case, we do not have to deal with replication, concurrency, dynamic partitioning of data, and crash recovery (see [15, 16, 8] for more details). In our simplified approach based on a **shared-nothing architecture**, we use a static and disjoint partitioning of the data in the cluster. Queries are sent to all nodes and the union of the resulting tuples is composed into the overall answer of the query. The partitioning is such that each node has to touch about the same amount of data for query evaluation. With TP-Lite, a query is executed in a master-slave setting using an instance of ORACLE8 on each node, its proprietary database links, and PL/SQL [13]. The coordinator runs both the master and slaves in independent transactions and we use ORACLE pipes as their communication primitive. A SQL-query is executed as follows: the master sends a message to all slaves over ORACLE pipes to initiate query execution. Each slave executes the SQL-query against the database of a dedicated node in the cluster using a database link. The result tuples are sent back to the master as messages over a shared ORACLE pipe. TP-Lite can be seen as a poor man's implementation of a parallel database. Although Böhm et al. [3] reported that TP-Lite is easily outperformed by using a TP-monitor or a proprietary coordination layer, we have used the TP-Lite implementation in conjunction with OLAP on clusters of commodity PCs to demonstrate the ability of our performance monitoring framework to detect and identify performance bottlenecks. As seen in Section 4, our methodology is able to document and explain why the TP-Lite approach is not always scaling well when increasing the machine size to a larger amount of cluster nodes.

2.3 The TPC-D Benchmark

The TPC-D benchmark [21] comprises a broad range of decision support applications that require complex, long running queries against large data structures. We are not interested

in publishing new results for a benchmark that is obsolete since 1999, but in demonstrating how our particular performance monitoring framework detects resource bottlenecks and architectural problems in parallel architectures.

The TPC-D benchmark contains 6 dimension tables (Customer, Nation, Region, Supplier, Part, PartSupp) and 2 fact tables (Order, LineItem). Out of the 17 queries of the TPC-D benchmark, we mainly used query 1, 3, 4, 8 and 12 for the experiments since they read large parts of the fact tables. As mentioned above, our implementation of a parallel database uses static partitioning of data. For space optimization reasons, we have partitioned the data of the fact tables, which coincide with the larger tables too, and we have disjointly distributed their data over the nodes of the cluster, while the dimension tables are fully replicated on all nodes. Since the queries under consideration run against large parts of the fact tables, we achieve a speed-up with the cluster simply due to the reduced volume of data touched by each node.

2.4 Experimental Setup

Clusters of commodity PCs with off-the-shelf software are becoming the most popular platform for very large data processing tasks. Therefore, we look at the execution of a parallel query on top of multiple instances of ORACLE running on nodes of a cluster of PCs under the LINUX operating system.

We refer to the database software needed to do this task as DBMS. An application-run in our experiment depends on several parameters controlling the workload and the execution environment. The parameters under investigation are called factors. Most factors are external and under the control of the application writer. However, our approach also deals with parameters that affect performance but are not directly visible to the application writer such as the set of factors related to the characteristics of the platform (i.e. the architectural parameters of a PC cluster) that we define the *platform factors*. Among them we consider the factors at the following levels:

- the clock rate of the CPU (t_{clock_cycle}): 400 MHz Pentium II (Deschutes) or 1000 MHz Pentium III (Coppermine),
- the average disk read performance¹ ($disk_rate$): 22.0 MB/s (slow disk) or 30.5 MB/s (fast disk)
- the average disk access time (t_{rand_access}): 7.3 ms (slow disk) or 6.8 ms (fast disk),
- the speed of the network interconnect ($network_speed$): 100 Mbit/s (Fast Ethernet) or 1000 Mbit/s (Gigabit Ethernet),
- the number of slaves in the cluster (ns): 1, 3 or 6 processing nodes².

¹The disk characteristics in detail are: slow disks, Seagate SCSI ST318203LW: with a min/avg/max throughput of 14.5/22.0/26.9 MB/s and access time 7.3 msec, the fast disks are Seagate SCSI disks ST318404LW with a throughput of 22.8/30.5/36.3 MB/s and 6.8 msec access time.

²The power database project aims at the scalability to a larger number of nodes (16, 32, or 64 processors), however measuring such task requires very large data sets. With

Besides the factors (parameters under investigation), some parameters are held at constant level for this investigation. For the sake of better explanation, they are written as variable in the analytical performance model of the performance monitoring framework. Among them are the clock cycle per instruction ($CPI = 1$) linking CPU clock-frequencies to integer performance, the size of a memory block ($blk = 512$), which is an important constant in the I/O buffering system of LINUX, the number of disks per node (3 disks: one for the system information, one for the indexes and a third for the data), the physical capacity of each disk ($disk_size = 18$ GB), the total size of the tables in the TPC-D database benchmark (10GB) and the memory size (512MB). The main motivation behind the migration of OLAP database work to clusters of PCs is to run huge databases of terabyte-scale on PC clusters that are equipped with a large number of inexpensive disks.

3. PERFORMANCE MODELING FRAMEWORK FOR PERFORMANCE EVALUATION IN CONTEXT OF DISTRIBUTED SYSTEMS

3.1 Structure of the Performance Modeling Framework

The raw data provided by the low level performance monitoring hardware must be filtered and properly abstracted to become useful for the analysis of a database system. To address this problem we propose and implement a performance monitoring framework that is suitable for a parallel computing environment. The framework comprises software instrumentation at the OS level, performance data gathering tools and an analytical model that is tailored to a particular application. We suggest that the performance modeling framework is developed separately - side by side to the DBMS middleware - and maintain that its functionality should not be integrated with the vendor specific DBMS, which we consider as a black-box. The software system of our performance modeling framework is structured into three different layers: an application-specific layer, a distribution-specific layer and a system-specific layer.

The *system-specific layer* of performance modeling framework monitors and collects system-specific performance data. System-specific performance data includes information on resource usage and bottlenecks gathered over the lifespan of the application-run. In our current prototype, we monitor the following resources: the local CPUs usage, the local disks usage and the local network usage as sampled on each platform node.

The *distribution-specific layer* gathers the performance related data from several nodes and patches it into a single coherent view of the whole system to be handed over to an application-specific layer. The performance modeling framework inherits the master-slave setting from its DBMS counterpart. The information gathered by the master is already properly filtered by the slaves and ready for processing at

our current experiments, we use 10 GB of data which is adequate for 1, 3, 6 processors. We plan to extend our work to the new 128 node "Xibalba" database cluster as soon as we manage the engineering challenge of generating and distributing data sets of 100 GB to 10 TB size with LINUX.

the application-specific layer.

The *application-specific layer* uses the detailed knowledge about the resource usage which is finally combined into a global performance picture describing the entire system and permitting high level optimizations. The top-most layer uses an analytical performance model of the application to map the resource usage to a suitable level of abstraction for performance tuning.

The issues related to the decomposition of the performance modeling framework into layers remains beyond the scope of this paper are addressed in detail in [19].

3.2 Some Critical Issues Addressed in the Performance Monitoring Framework

We consider a distributed system running a parallel application code over an extended amount of time. Therefore, we have to cope with a large amount of performance related information, which we call response variable according to [11]. An complete representation of all relevant performance data is rather expensive, in particular if the distributed system has a large number of nodes to be monitored. A huge amount of information would have to be exchanged within the distributed performance modeling framework layers leading to a high overhead, unless the amount of system information is filtered. Instead of recording every event possible, some parameters must be captured by sampling. If samples are not taken frequently enough, it is impossible to make an accurate statement about the system, but if they are taken too frequently, the system is perturbed by floods of monitoring traffic. The **granularity of samples** must be a trade-off between accuracy of the system view and the cost for the system monitoring. Compression into a representation of minimal size and appropriate granularity is necessary. In our experiments, we refer to a local sample every one second.

The performance data (response variables) we collect from the nodes comprises:

- the number of instructions (i.e. user instructions $IC_{user}(j)$ and system instructions $IC_{sys}(j)$) on the coordinator and nodes CPUs. j ranges from 1 to $ns + 1$ where ns is the factor expressing the number of nodes.
- the number of sequential disk accesses ($seq(j)$) and non sequential disk accesses ($no_seq(j)$) to the disk of the coordinator and each node (j ranges from 1 to $ns + 1$).
- the average stride $avg_stride(j)$ for the disk access in each node and the coordinator. By average stride, we relate to the average movement of the disk head between two random accesses to a disk. Sequential disk accesses show up as reads with a stride of one while non sequential accesses show up as larger strides¹.
- the amount of traffic transferred over the network interconnect (i.e. Fast Ethernet, Gigabit Ethernet). We distinguish between the amount of *bytes received* ($size_rec(j)$) by each node and the coordinator on network device, and the amount of *bytes sent* ($size_trans(j)$) from each node as well as from the coordinator on network device. Again, j ranges from 1 to $ns + 1$.

¹In our model the block index actually increments by two since LINUX always read two blocks (blk) at a time.

One important point in the design and development of our performance monitoring framework is how deeply the system-specific layer should be integrated with the operating system. Our best solution is to implement the system-specific layer of performance modeling framework outside the kernel as a daemon. This renders monitoring slightly slower and less accurate than in a kernel implementation, but remains much easier to maintain with new version of the OS kernel (which happen rather frequently). The prototype is based on LINUX */proc file* mechanism to write performance data. However, the information available in the */proc file* are significantly extended. The framework of performance modeling framework uses the *hardware performance counters* of the Pentium processor [10] that are made accessible through a library we have implemented ourselves. All information for performance analysis is gathered at every node of our distributed system in parallel and the performance modeling framework puts it all together into a global view of the parallel system. The monitoring tools in the nodes send the performance information to the monitoring master for each sample. The **network overhead** for these frequent data transmissions is kept low using the UDP/IP protocol, which is known to work quite well in clusters of PCs where links are short, full crossbar switches are common and transmission errors are infrequent. Many programmers are still building distributed monitoring tools with TCP/IP involving retransmission and flow control. We learned that for the best sampling accuracy and for the least system disturbance it is better to use UDP/IP and deal with the loss of information. Such loss of messages can be treated like sampling errors. Moreover, the use of the UDP/IP protocol optimally fits our specific **notion of time** within the distributed system. Communicating monitoring data through TCP/IP would result in unwanted synchronization by the TCP flow control messages, which add unacceptable overhead and perturbation to our applications. Maintaining a consistent notion of time in a distributed system is an important aspect of our performance monitoring framework. The monitoring master has to be able to patch together the performance data of the several nodes in a consistent manner with a global wall clock time scale. To address this issue, a consistent, single notion of time among the several nodes of the distributed system has to be maintained. To overcome this problem, the monitoring framework has to introduce some mechanisms of synchronization during the performance sampling without introducing additional monitoring intrusions. To cope with the problem of maintaining a global notion of time and at the same time reducing monitoring intrusions, we propose a notion of time based on accurate built-in cycle counters. We avoid unnecessary synchronizations through a highly precise synchronization at the start of the monitoring session and, as the execution progresses, relaxing to a loose synchronization [7] in which the monitoring tool synchronizes sampling by looking at the highly accurate cycle counters in the CPUs. The built-in cycle counters mechanism acts as *virtual barriers*. The cycle counters have to be delivered as time-stamps in the performance packets containing the sample information sent to the monitoring master.

3.3 Performance Modeling

The application-specific layer of performance modeling framework includes a performance model of the application that can translate the elementary knowledge about the resource

usage into high level answers to performance questions and performance bottleneck that are suitable for suggesting optimizations to the user.

Our current model is a simple set of formulas which allow the calculation of the individual execution times due to the usage of each machine resource, such as CPU, disk or network.

Table 1 shows the set of equations making up the analytical model used for the estimates of total execution time based on data about the detailed resource usage. The model aggregates low level response variables as an effect of the factors chosen in Section 2.4. Since we are dealing with resource constraints, the entire table talks about max values of the time components given by the several nodes and the coordinator. Note that the time component of CPU usage is directly related to the amount of instructions on the nodes, the time for the disks usage is the sum of the time for sequential and non sequential accesses to the disks while the communication time depends on the communication rate and message size both gathered through the performance modeling framework in the master-slave setting.

4. PERFORMANCE ISSUES STUDIED WITH OUR FRAMEWORK

To demonstrate our novel monitoring framework at work, we look at three important performance issues including the workload characterization of the TPC-D queries, their scalability and the dependency on the network interconnect in a cluster of commodity PCs. We present some solid explanations and remedies for the resource bottlenecks and performance problems found.

4.1 Workload Characterization of Distributed TPC-D

The TPC-D benchmark uses 17 different queries that fall into a few different categories. As database specialists, we would typically classify them into categories based on the number and the extend of join operations required to work through the relation tables affected by each query. Our current approach as system architects and performance evaluation specialists is quite different; we propose an alternative classification of the queries based on the most critical machine resources (i.e. CPU, disks, and network) used during their execution. We look at one single query at the time and run it in parallel on a share-nothing architecture in which each machine consists of a processor, a memory and three disks. The nodes communicate to each other by means of high-speed interconnection networks [17].

The different machine resources in our distributed system (local or remote) differ a lot in speed and availability. In our performance model that is part of the performance monitoring framework, we identify three different machine resources that can be a critical limit to faster execution of a query: the CPU usage, the disk usage and the network usage for inter-node communication in the case of parallel processing. We consider CPU, disk and network usages because of their direct significance to the database applications. Moreover, we add a fourth category for queries whose performance can not be modeled accurately based on resource usage because revealing an obvious inefficiency in one of the DBMS layers. This approach and the related set of resources is certainly not limited to database applications. We successfully use

resource	estimated time	variable factors	response variables
CPU	$\max_{j=1..ns+1} (CPI * (IC_{user}(j) + IC_{sys}(j)) * t_{clock_cycle}(j))$	$t_{clock_cycle}(j)$	$IC_{user}(j), IC_{sys}(j)$
disks	$\max_{j=1..ns+1} (t_{seq}(j) + t_{no_seq}(j))$ $t_{seq}(j) = \frac{2 * blk * seq(j)}{disk_rate(j)}$ $t_{no_seq}(j) = (no_seq(j) * t_{avg_stride}(j))$ $t_{avg_stride}(j) = \frac{(blk * avg_stride(j))}{disk_size(j)} * t_{rand_access}$	$disk_rate(j)$ $disk_size(j)$ $t_{rand_access}(j)$	$seq(j)$ $no_seq(j)$ $t_{avg_stride}(j)$
network	$\max_{j=1..ns+1} (\frac{size_rec(j) + size_trans(j)}{rate_rec(j) + rate_trans(j)})$		$size_rec(j), size_trans(j)$ $rate_rec(j), rate_trans(j)$

Table 1: Set of equations making up the analytical model used for the estimates of total time for the usage of the resources chosen for the OLAP application.

a similar method for the analysis and modeling of parallel scientific codes [18].

A typical example of a query limited by CPU usage is query 1, a typical example of a query limited by disk usage is query 4. Query 3 exhibits an interesting dependency on the communication subsystem when executed on multiple nodes and is therefore a good example for a query limited by network usage. Finally, query 8 shows the limitation of our performance evaluation techniques since its resource usage appears to be totally inconclusive. Unlike in all other cases where the processed output of our monitoring tools and our analytic model explain the execution time within an average error of less than 10%², in this case most of the execution time remains without direct allocation to machine resources in the system.

Figure 2.a (left) breaks down the usage time allocated to the specific machine resources by the performance modeling framework for query 1 with two different generations of PC clusters that differ in the CPU clock rates. We break down the execution time into four components: CPU usage, sequential access and non sequential access to the disk subsystem and inter-node communication. Figure 2.b (right) shows the percentage of these components. The performance modeling framework shows that the CPU usage accounts for 80% to 60% of the execution time, depending on the clock rates of the CPUs used. The component of the execution time attributed to CPU usage diminishes by a factor of 2.5 when we upgrade the CPUs from 400MHz to 1 GHz proving that query 1 is largely CPU limited. Looking at the scalability with a growing number of nodes, we state that the fractions of execution times stay the same and that the limiting factor remains the CPU even for larger clusters. The precise extent of scalability will be discussed in the next subsection.

Figure 3.a (left) breaks down the usage time allocated to the specific machine resources by our performance modeling framework for query 4 with two different disk types used in the nodes of our clusters. Again, Figure 3.b (right) shows the percentage of the time components. The data provided by the performance modeling framework indicates that the disk usage accounts for more than 90% of the execution time in both cases of disk subsystems. The performance monitoring framework is further capable to distinguish sequential from non sequential (random) read disk accesses. As expected for OLAP workload, the emphasis is on sequential read access with a ratio of 90%/10% for the slower disks and 85%/15%

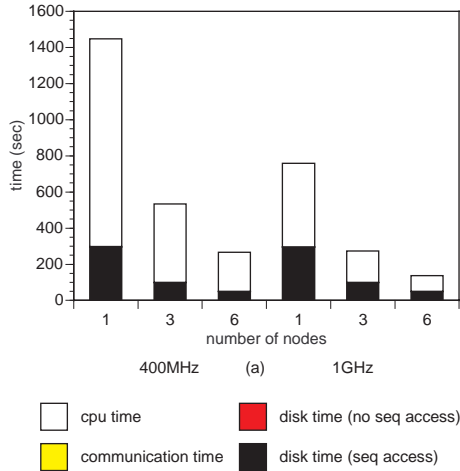
²We did calibrate the analytical model with numerous measurements and we do have the data to show its accuracy, but we had to omit the figures due to space constraints.

for the faster disks. Going to faster disks does significantly decrease the execution time as can be seen in Figure 3.a. The fraction of execution time allocated to disk operations by the performance monitoring framework decreases with faster disks as expected. As for scalability with 1, 3 or 6 nodes, the fractions of CPU-, disk- and network usage stay the same for all distributed configurations.

We identified query 3 as a representative of a network-limited query. Figure 4.a (left) shows the usage time allocated to the specific machine resources by the performance modeling framework for query 3 (i.e. CPU, disk and network usage) with two different network interconnects commonly found in clusters of PCs (i.e. Fast Ethernet and Gigabit Ethernet). As expected, there is no inter-node communication in the uniprocessor case (1 processor) and the communication becomes only visible as we distribute the workload to multiple nodes (3 and 6 nodes) in the PC cluster. The distribution of the resources without the network is about 30% CPU and 70% disk and stays that way for larger clusters pointing at almost linear scalability for the CPU and disk components and at the good explanation of the non scalability by the communication work. Surprisingly, there is no improvement with the addition of Gigabit Ethernet that is 10 times faster than Fast Ethernet. The scalability and the impact of the network will be discussed below - for now we just state the evidence that query 3 is a network-limited query in parallel OLAP on clusters of PCs.

We classify the queries whose total usage time allocated to the specific machine resources by the performance modeling framework for is much shorter than the total run-time measured (less than 15% in average) as DBMS inefficient or DBMS-dependent. The performance monitoring framework reports that for such queries, none of the monitored resources are intensively used. The lack of a critical machine resource indicates an internal problem in the DBMS. The non-allocated time is due to an ill-conceived sequence or amount in which the different resources are demanded by the DBMS. Certainly, there might be an additional non-monitored resource or a combination of resources that could be the bottleneck, but we did not find any and it remains unlikely that the entire set of monitoring tools of a modern operating system would not record any indication. In parallel scientific codes, we found such behaviors due to load imbalance and due to ill fated synchronization algorithms [18]. Query 8 is characterized by such a behavior. Figure 5.a (left), indicates the time allocated to the resource usage by the performance monitoring framework and the non allocated time for this query. We can see that the phenomenon

Components of time allocated to the resource usage for query 1 with different CPU clock rates and number of nodes (fast disks, Gigabit Ethernet)



Percentage of time components allocated to the resource usage for query 1 with different CPU clock rates and number of nodes (fast disks, Gigabit Ethernet)

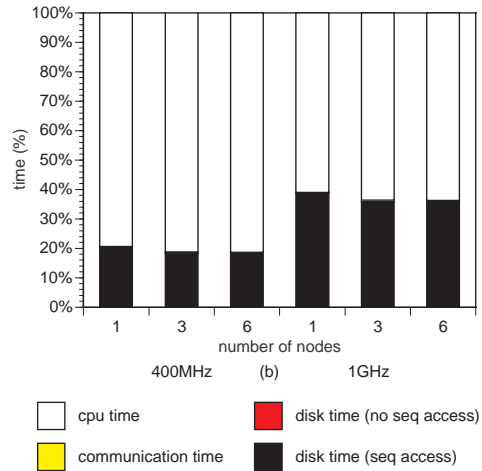
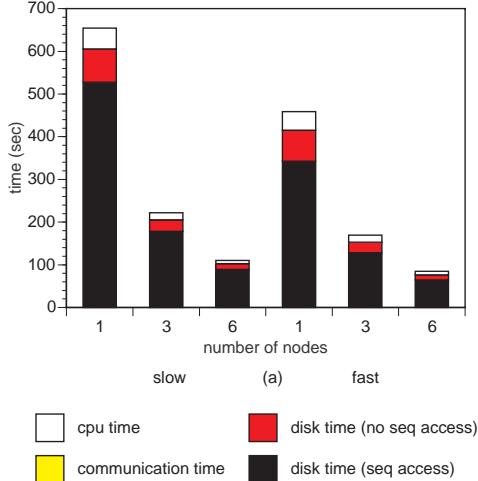


Figure 2: Components of the execution time allocated by the performance modeling framework to the resource usage (CPU, disk and network) for query 1 in seconds (a) and as percentages (b) for different CPU clock rates and different number of nodes. CPU usage for query 1 accounts for 80% to 60% of the execution time, depending on the clock rates of the CPUs used. Network usage and non sequential accesses to the disks are negligible.

Components of time allocated to the resource usage for query 4 with different disk speeds and number of nodes (1 GHz, Gigabit Ethernet)



Percentage of time components allocated to the resource usage for query 4 with different disk speeds and number of nodes (1 GHz, Gigabit Ethernet)

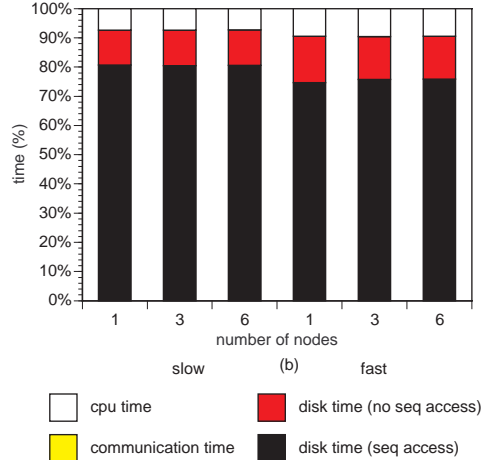
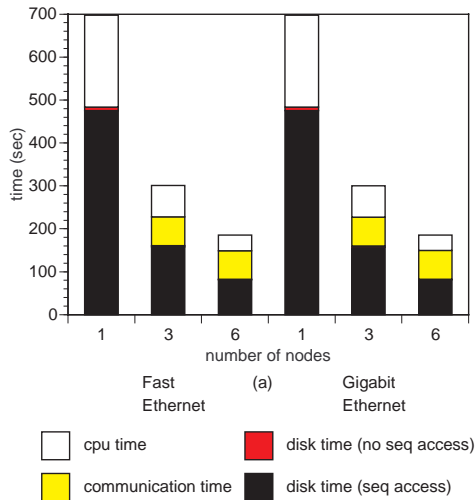


Figure 3: Components of the execution time allocated by the performance modeling framework to the resource usage (CPU, disk and network) for query 4 in seconds (a) and as percentages (b) for different disk speeds and different number of nodes. Disk usage accounts for more than 90% of the execution time in both cases of disk subsystems.

is present to the same extent in the centralized case (1 processor) as well as in the parallel cases (3 or 6 nodes). The time spent based on the profiled workload of the CPU, disks and network covers less than 20% for the query running on a single node and increases only slightly to over the 25% for six nodes as indicated in Figure 5.b (right figure first three bars). A further breakdown of the time explained by the performance monitoring framework in query 8 is a bit incon-

clusive. The part of query 8 we can explain seems to be disk limited with a large emphasis of random disk accesses. In Figure 5.c, it also seems that the sequential accesses and the CPU usage does scale with a larger number of nodes while the non sequential disk accesses do not scale (right figure last three bars). Such a situation indicates a high potential of optimization through tuning parameters of the application code or the DBMS. In fact, we have access to a few intrinsic

Components of time allocated to the resource usage for query 3 with different networks and number of nodes (1 GHz, fast disks)



Percentage of time components allocated to the resource usage for query 3 with different networks and number of nodes (1 GHz, fast disks)

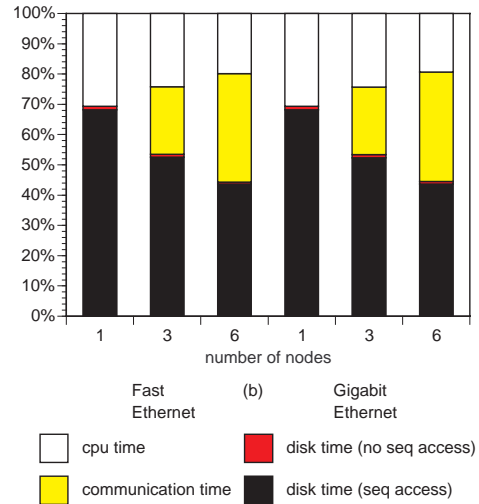
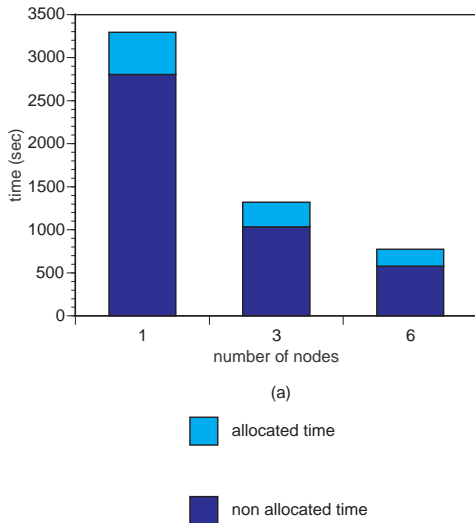


Figure 4: Components of the execution time allocated by the performance modeling framework to the resource usage (CPU, disk and network) for query 3 in seconds (a) and as percentages (b) for different networks (i.e. Fast- and Gigabit Ethernet) and different number of nodes. The fraction of communication time increases with the number of nodes (bad scalability). After deduction of the network part, the 30/70 ratio between CPU and disk remains invariant regardless the size of the cluster.

Allocated and non allocated time to the resource usage for query 8 with different number of nodes (1 GHz, fast disks, Gigabit Ethernet)



Percentage of allocated and non allocated time to the resource usage (b) and related time components of the allocated time (c) (1 GHz, fast disks, Gigabit Ethernet)

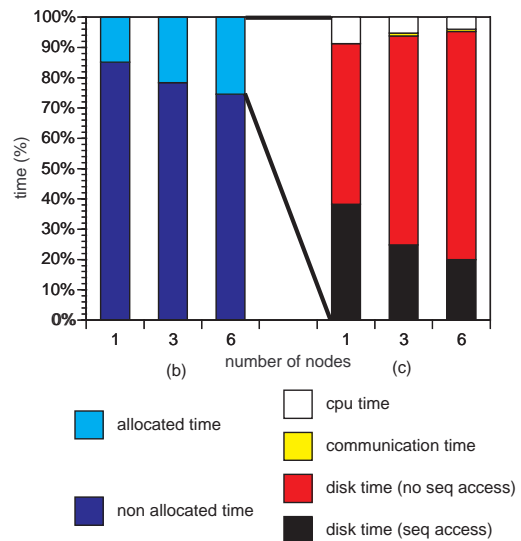


Figure 5: Execution times that *can* be allocated to the use of specific machine resources and time that *cannot* be allocated by our performance monitoring framework in query 8. The time is given in seconds (a) and percentages (b). The allocate-able part is further broken down into resource usage as observed by the performance monitoring framework (c). None of the monitored resources are intensively used. This may be due to ill-conceived resource demands by the DBMS (i.e. the sequence and the amounts in which the different resources are demanded).

parameters of the system that affect the performance. In the TP-lite approach the distribution of the relation tables

can be set to a more or less aggressive strategy for partitioning vs. replication. We can use our performance monitoring

framework to determine the best partitioning by parameter variation [19].

The performance monitoring framework with the collected performance monitoring information combined with an analytical model of resource usage permits to characterize the workload of TPC-D OLAP application more closely through a classification of the queries according to their individual resource usage. With the help of our framework, we can classify the queries as CPU-limited, disk-limited, communication-limited or inefficient due to DBMS limitations.

4.2 Scalability of Parallel TPC-D in a Cluster of PCs

Figure 2.a and Figure 2.b show that CPU usage is the most important execution time component in the processing of query 1 and that this component scales almost perfectly with a growing number of nodes involved in a parallel processing of this query. The second most important resource limitation is disk usage. Since we use a shared-nothing architecture to distribute the workload, the number of disks increases with the number of nodes involved. The inter-node communication stays negligible in this query since very small amount of data is transferred. Figure 3.a and Figure 3.b confirm a similar picture for query 4. This query results in heavy on disk usage. Again this is invariant to changes in the number of nodes because the total disk performance scales with an increasing number of nodes. The second most used resource is CPU usage which scales perfectly to larger systems.

For query 3, the network usage does no longer scale nicely with the increase of the number of slaves, while the other resources (i.e. CPU usage and sequential access to disk) have good scalability (see Figure 4.a and Figure 4.b). The reason for the limited scalability is a growing percentage of execution time due to inter-node communication with 3 and 6 nodes involved. Furthermore, the time components scale exactly in the same way for 1000 BaseT and 100 BaseT, and therefore, the loss of scalability seems to be independent of the strength of the interconnection network.

The amount of communication required to process a parallel query is not necessary a reason for bad scalability. In our clusters of PCs, the nodes are interconnected with a full crossbar switch and therefore, the network resources available to the parallel processing of OLAP workload could actually grow linearly with the number of processing nodes involved. Furthermore, the total amount of communication data reported by the monitors and collected by the performance monitoring framework is fairly small and does not point directly at a bottleneck. Even a slow, shared medium Fast Ethernet (hub) would be able to provide the necessary raw bandwidth to move the data. Therefore, a proper communication performance study requires more than the total amount of data transferred.

The global sampling of the communication data by means of the performance monitoring framework is accurately synchronized by a virtual wall clock time and permits to identify bursty and unbalanced network traffic in the parallel application.

Figure 6 on the left shows the communication activity on the coordinator node for query 3 while coordinating the processing of the query with three/six nodes (coordinator/3 and coordinator/6). At the same time the figure displays on the right the communication activity for the same query

on the three/six individual processing nodes (node n/3 and node n/6). The data in the pictures are gathered at the distribution-specific layer by the performance monitoring framework.

In the top-right charts we consider the communication on each of the three nodes for the processing of the query with three nodes, while in the bottom-right charts we look at the communication on the first and last nodes for the processing of the query with six nodes. The charts related to the processing on three nodes (top) depict the transferred number of bytes per second as a function of time where 0s denotes the start of query execution and 280s the end of query execution for three nodes. The same query finishes earlier i.e. roughly at time 170s when the work is distributed among six nodes (see charts on the bottom). In the left charts we look at the communication work on the coordinator and distinguish between “sends to the nodes” and “receives from the nodes”. In the right charts we distinguish between “sends to the coordinator” and “receives from the coordinator”. The graphs show a network traffic that is quite bursty and unbalanced, most of the traffic is concentrated on the coordinator, while the processing nodes clearly communicate at roughly one third or one sixth of the speed of the coordinator.

The graphs refuted our initial assumption that communication is highly asymmetrical due to the algorithm in TP-Lite that processes partial queries on each node and finally gathers the results in the coordinator. In reality, the communication traffic is quite symmetrical and during the data transfer the coordinator sends almost as many bytes of request as the nodes sends for its answers. Furthermore, the peak communication rates in the coordinator is determined to be just 6.4 MBit/s in a network that can sustain one Gigabit/s (i.e. 160 times more) under good conditions.

The performance monitoring framework provides precise allocation of total execution time to each resource usage and offers the possibility to record resource usage for arbitrary sampling intervals. This helps to find and to isolate the cause for the loss of scalability in the processing of non scalable queries. An accurate sampling of all communication activity over the entire execution time is required to discover and deal with performance limitation due to bursts and hot spots in the communication patterns.

4.3 Performance Impact of the Network Interconnect Speed

During the evaluation, procurement and installation of a large cluster for database research, we looked at the least expensive interconnect technology that is sufficient for the planned work on parallel databases. For a Beowulf type cluster, a commodity Fast Ethernet switch would be sufficient. For enhanced clusters, we would opt for a Fast Ethernet switch that provides full bisection bandwidth (or non blocking ports in networking terminology). For an advanced cluster, we consider high performance interconnects like Gigabit Ethernet or Myrinet. Depending on the performance networking, a cluster node can cost between US\$ 100 and US\$ 1000 per node installed. Our study should answer the question of whether clusters with higher interconnect speeds are worthwhile for parallel OLAP processing or not. For our measurements, we equipped our cluster of PCs with Gigabit Ethernet and a fully non-blocking 16 port switch in addition to non-blocking Fast Ethernet.

The study of the benefit of a higher speed interconnect is limited to query 3, an example query that obviously becomes

Communication speed of query 3 on coordinator and each of three/six processing nodes connected by Gigabit Ethernet

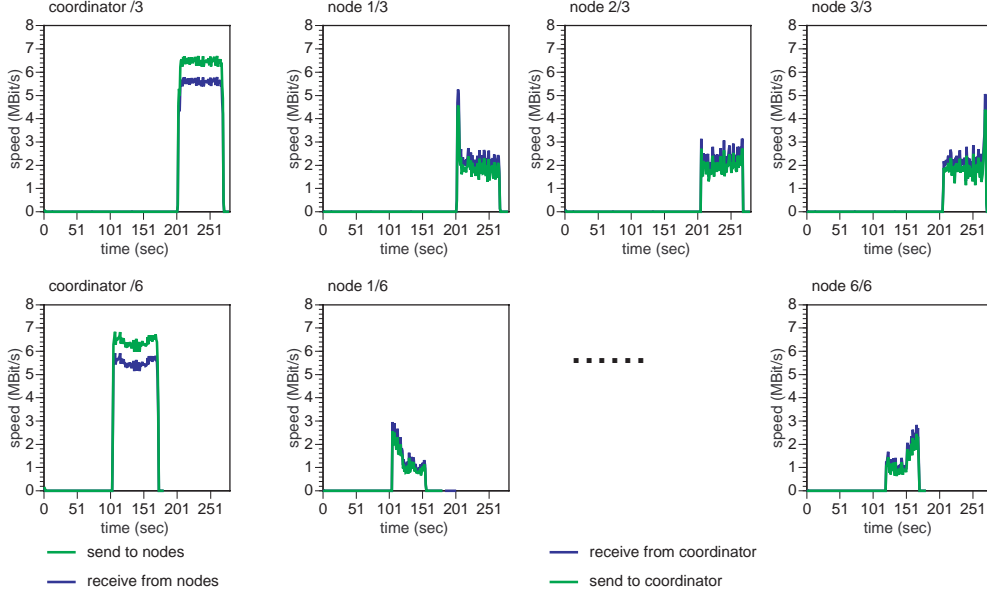


Figure 6: Communication activity during the the processing of query 3 for three/six nodes on the coordinator node (left) and on each of the three/six processor nodes (right).

communication bound in the parallel configurations of 3 or 6 nodes. As seen in the Figure 6, the communication activity is bursty and takes place towards the end of the query. Furthermore, the communication takes place between the coordinator and all the nodes at the same time (two peaks, one for the 3 node experiment and another for the 6 node experiment), but most importantly the speed of communication is just 6-7 MBit/s which is two order of magnitudes below the 100 MBit/s that is possible with Fast Ethernet and three order of magnitudes slower than Gigabit Ethernet. It is not visible from the chart if the inefficiency is due to further burstyness or packet collisions within the sampling interval. Alternatively, software could be responsible for the bad performance. A separate experiment must be used to verify if there is any benefit at all in using a faster networking technology (which would handle bursty traffic faster).

Figure 7.a and Figure 7.b depict the trace of communication activity on a node (transfers from and to a coordinator) for query 3 in a system of three nodes of cluster connected by Fast Ethernet and Gigabit Ethernet respectively. The two nearly identical network usage traces prove that communication behavior is exactly the same for a 100 MBit and a 1 GBit networks. Therefore we see no benefit in purchasing a network with a higher throughput.

In our first approach for a parallel processing of OLAP database workloads, we used the mechanism of ORACLE database links to ship data from the nodes to the coordinator in the cluster. This mechanism is based on the ORACLE NET8 transport protocol and uses 2PC (two phase commit) to ensure transactional guarantees. Our experiments indicate it is not efficient enough to work on large amounts of communication since it requires a lot of slow synchronization and is badly affected by high network latency or large processing overhead. Moving the application to clus-

ters with low-latency networks (like e.g. Myrinet) could still fail to solve the communication problems due to high software overheads within the DBMS. And even if would help, it is most certainly not cost-effective since the bandwidth of Myrinet will remain unused most of the time.

Again, the performance monitoring framework with its accurate sampling and its global representation of resource usage in the distributed system permits an in-depth analysis of communication problems and properly explains the inefficient use of the network while processing queries like query 3 in TPC-D. With our performance analysis framework, we are finally able to explain why the TP-Lite approach behaved so much worse than the other approaches in [3].

5. ARCHITECTURAL IMPLICATIONS OF THE RESOURCE USAGE OBSERVED

Following the classification of the queries reported in the previous section, we can calculate the CPI value from the measured performance data for the different classes of TPC-D queries and find numbers that are quite interesting. CPU limited queries are indicative for the basic CPI value of OLAP workloads, but since we are on a parallel and distributed system we are experiencing several modifications to those basic values. If the nodes processing a query spend a lot of time waiting for “peripherals” such as disks (SCSI cards) or the network (Ethernet cards), the cycles are not attributed to processing instructions but to idle loops in the operating system kernel. Therefore, the CPI is inflated as soon as disk or network do most of the work and are frequently waited for.

For strictly CPU-limited queries (e.g. query 1), we measure a CPI of about 1.5 on the older and the newer types of

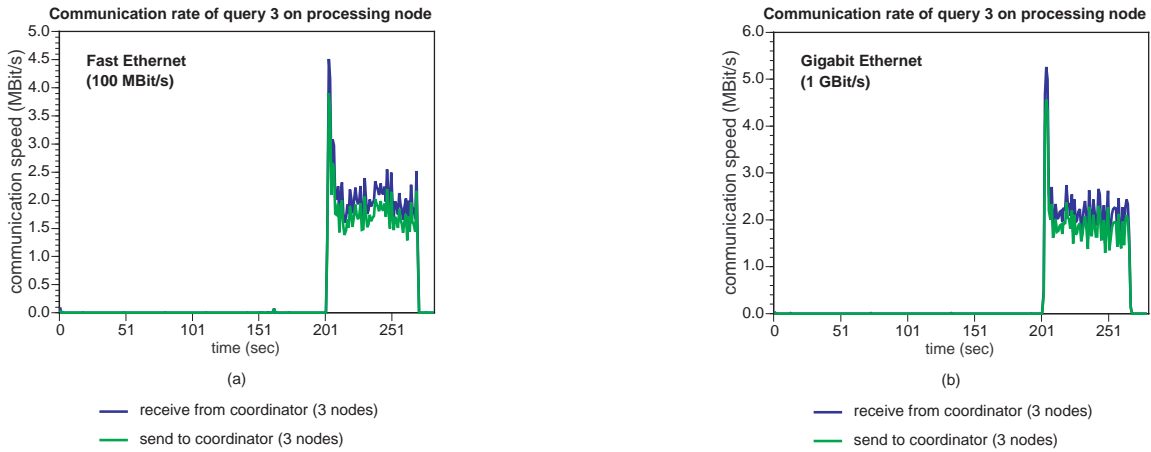


Figure 7: Slave communication rates of query 3 for Gigabit Ethernet (a) and for Fast Ethernet (b) on a cluster with three slaves and CPUs with 1 GHz clock rate.

nodes, regardless of clock rate of either 400MHz or 1GHz. Researchers of UCB and UIUC looked at CPI figures for TPC-C and TPC-D on large symmetric shared-memory multiprocessors (SMPs). We are not aware of any studies that deal with massively parallel supercomputer nodes or entire clusters of PCs up to this date.

Keeton *et al.* [12] look at the CPU usage of the TPC-C benchmark using Informix DBMS running on an Quad Pentium Pro, shared-memory, shared-disk configuration. The basic CPI including memory usage is 2.90, which is much higher than the CPI values found in computational benchmarks like SPEC95. Those values are for OLTP and so the difference to our values of 1.4 to 1.5 for OLAP is not surprising. Cao *et al.* [5, 4] look at the characterization of the TPC-D benchmark with Microsoft’s SQL Server and Windows NT on top of SMP server with Quad Pentium Pro, shared-memory, shared-disk configuration. They report a basic CPI of 1.27. This slightly lower figure is for a high end SMP with moderate parallelism, while our work deals with cluster nodes in a massively parallel setting.

We carefully study the dependency the CPI on the different queries, i.e. the workload. For disk-limited queries (e.g. query 4), the CPU is frequently waiting for the disk accesses, which inflates the overall CPI considerably. The CPI measured on clusters with the faster disks is about 10 while the CPI for slower disks is about 15. Communication-limited queries are unique for parallel and distributed computing platforms. The resulting CPI values of 3 to 4 reflect the idle loops encountered for waiting for data from the network. Finally, the DBMS software limited queries show extremely high CPI values that can only be interpreted by a failure of the monitoring environment to track down the resource usage properly. Query 8 is characterized by a CPI of about 90.

The good values of the CPI for CPU and disk limited queries indicates that a cluster architect should focus on purchasing fast disks and purchasing fast CPU. This is consistent with the view of processor architects that see database workloads as a major driver for ever better higher MIPS ratings in microprocessor with higher clock rates and more ILP in those database workloads [2].

The performance picture of our study turned out to be dev-

astating to high speed communication in clusters of PCs, which was a primary target of the clusters built in our CoPs (Clusters of PCs) project.

Distributing OLAP queries to a large number of nodes with partitioned data can result in large amounts of inter-node communication for certain queries. Unfortunately, the communication is completely dominated by software overheads within the DBMS and adding a faster network does not help. More precisely neither better latencies nor better bandwidth results in much improvement. The precise analysis of the network usage over time indicates that there is a bottleneck due communication, which takes place between the coordinator and all the nodes at the same time. The bottleneck can be removed by improving the scheduling of the partial queries and by balancing the load communication more carefully. As expected for a database application, the performance of the disks is highly critical to overall performance. Using a distributed file-system with RAID capability based on remote disk accesses may improve the OLAP performance beyond the limitations of simple parallelization.

6. CONCLUSION

Performance analysis in parallel databases running on clusters of commodity PCs remains a highly difficult task, since we are still lacking many of the tools and instrumentation that could give us the performance data we need to understand parallel- and distributed systems executing OLAP workloads in standard DBMSs.

As a contribution to address this important problem, we present a performance monitoring framework for collecting and filtering the raw performance data given by the operating system in a distributed high performance database system built from common commodity PCs and commercial DBMSs. Our framework for performance monitoring combines the usual monitoring instruments at the operating system level with an effective strategy for collection and interpretation of this monitoring information at the overall systems level. Together with some simple analytical model of overall resource usage, we can assess the high level performance indicators at a level of abstraction that is appropriate

for an application writer. In particular, our system is able to give some new insights about the use of specific machine resources in the system. In the existing monitoring system, we capture CPU usage (CPI), two kinds of disk usage (sequential and non-sequential), communication system usage and the work for memory system usage is currently progress. Unlike the built-in performance monitors of most database management systems, our operating system based monitoring solution is fully operational in a cluster setting with distributed processing and accounts for accumulated resource usage as well as for a time-variant resource usage by sampling and collecting performance data at arbitrary intervals. Information about peak resource usage and temporary bottlenecks can be derived from the time-variant resource usage traces.

To demonstrate the viability of our approach and to draw proper architectural conclusion about the construction of future PC clusters for high performance database systems, we use our performance monitoring framework to investigate parallel high performance databases executing OLAP workloads on clusters of commodity PCs. We successfully analyze a highly complex hardware-software system that relies primarily on standard hardware and commercial software components that were provided to us by a database research group. The configuration evaluated includes an ORACLE DBMS for SQL processing at each node, LINUX operating system to manage the node's resources as well as different Ethernet switches to take care of inter-node communication. The experimental software shell to distribute the queries to different nodes (TP-Lite) is taken from a database research project.

In our measurement effort, we execute a 10GB TPC-D benchmark and to characterize the workload precisely according to the resource usage encountered in the cluster. We gain significant insight into the question of the impact of high performance networking to this sort of application. As a most interesting result, we can classify the queries of TPC-D into CPU-limited, disk-limited or communication-limited queries including a fourth class that remains inefficient due to DBMS software limitations. Knowing the critical resource for each query we can properly explain scalability (or lack thereof) for each query on cluster with three and six nodes. Based on a simple analytic model that factors the total execution time into parts attributed to each resource, we can estimate scalability large number of nodes. To our biggest surprise, the inter-node communication is not the key to better scalability unless the communication facilities of standard DBMS software are drastically improved. In particular, we have shown that the Gigabit high speed network in our high end cluster of PCs does not improve scalability of the application due to inefficiencies in the commercial DBMS software that is used in conjunction with TP-Lite.

7. REFERENCES

- [1] A. Ailamaki, D.J. DeWitt, M.D. Hill, and D.A. Wood. DBMSs on a modern processor: Where does time go? In *Proc. of the 25th International Conference on Very Large Data Bases (VLDB)*, Edinburgh, UK, September 1999.
- [2] M.W. Blasgen. Personal communication.
- [3] K. Böhm, T. Grabs, U. Röhm, and H.-J. Schek. Evaluating the Coordination Overhead of Synchronous Replica Maintenance in a Cluster of Databases. In A. Bode, W. Karl T. Ludwig, and R. Wismüller, editors, *Proc. of the 6th International Euro-Par Conference*, Lecture Notes in
- Computer Science, pages 435–444, Munich, Germany, August 2000. Springer.
- [4] Q. Cao, J. Torrellas, and H. Jagadish. Unified Fine-Granularity Buffering of Index and Data: Approach and Implementation. In *IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'00)*, pages 175–186, Washington - Brussels - Tokyo, September 2000. IEEE Computer Society.
- [5] Q. Cao, P. Trancoso, J.-L. Larriba-Pey, J. Torrellas, R. Knighten, and Y. Won. Detailed Characterization of a Quad Pentium Pro Server Running TPC-D. In *Proc. of the International Conference on Computer Design (ICCD '99)*, pages 108–117, Washington - Brussels - Tokyo, October 1999. IEEE Computer Society.
- [6] C. Dye. *Oracle Distirbuted Systems*. 1st Edition. O'Reilly, 1999.
- [7] G.C. Fox. What have we learnt from using real parallel machines to solve real problems? In *Proc. of the 3rd conference on Hypercube concurrent computers and applications*, pages Vol.2 pp 897–955, Pasadena, CA, January 1988.
- [8] T. Grabs, K. Böhm, and H.-J. Schek. High-level Parallelisation in a Database Cluster: a Feasibility Study Using Document Services. In *Proc. of the International Conference on Data Engineering (ICDE)*, Heidelberg, Germany, April 2001. IEEE Computer Society.
- [9] M. Gurry. *Oracle SQL Tuning Pocket Reference*. O'Reilly, 2001.
- [10] Intel Corporation. Pentium Pro Family Developer's Manual, 1996.
- [11] R. Jain. *The Art of Computer Systems Performance Analysis*. Wiley Professional Computing, 1996.
- [12] K. Keeton, D. Patterson, Y. He, R. Raphael, and W. Baker. Performance characterization of a Quad Pentium Pro SMP using OLTP Workloads. In *Proc. of the 25th Annual International Symposium on Computer Architecture (ISCA-98)*, volume 26,3 of *ACM Computer Architecture News*, New York, June 1998. ACM Press.
- [13] Oracle Corporation. Oracle8. <http://www.oracle.com/>, 1997.
- [14] P. Ranganathan, K. Gharachorloo, S. Adve, and L. Barroso. Performance of database workloads on shared-memory systems with out-of-order processors. In *Proc. of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, California, October 1998.
- [15] U. Röhm, K. Böhm, and H.-J. Schek. OLAP Query Routing and Physical Design in a Database Cluster. In *Proc. of the International Conference on Extending Database Technology*, Konstanz, Germany, March 2000.
- [16] U. Röhm, K. Böhm, and H.-J. Schek. Cache-Aware Query Routing in a Cluster of Databases. In *Proc. of the International Conference on Data Engineering (ICDE)*, Heidelberg, Germany, Apr 2001. IEEE Computer Society.
- [17] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill International Editons, 1997.
- [18] M. Taufer and T. Stricker. Accurate Performance Evaluation, Modelling and Prediction of a Message Passing Simulation Code based on Middleware. In *Proc. of the SC98 Conference*, Orlando, FL, November 1998.
- [19] M. Taufer, T. Stricker, and R. Weber. Inverting Middleware Framework: a Framework for Performance Analysis of Distributed OLAP Benchmarks on Clusters of PCs by Filtering and Abstracting Low Level Resource Usage. Technical Report Technical Report 367, ETH Zurich, Institute of Computer Systems, Switzerland, March 2002.
- [20] The Transaction Processing Performance Council. TPC Benchmark C Standard Specification Revision 3.4. <http://www.tpc.org>, 1998.
- [21] The Transaction Processing Performance Council. TPC Benchmark D Standard Specification Revision 1.3.1. <http://www.tpc.org>, 1998.